User interest prediction for tweets using semantic enrichment with DBpedia

Denis Lukovnikov Mathias Verbeke Bettina Berendt

Department of Computer Science,
KU Leuven

Abstract

This paper focuses on topic-based prediction of interest of individual users to posts in the context of Twitter. Two methods for enriching tweets using DBpedia for the purposes of classification are proposed. The first method incorporates entity linking and uses linked entities in a tweet to improve classification, whereas the second method aims to improve upon the first one by adding information derived from DBpedia about entities found using the first method. The two methods are evaluated with respect to tweet classification.

1 Introduction

Interest classification of social items (e.g. tweets) is a challenging and relevant problem. Social network usage numbers already hint at a problem arising in our new information age. The problem is called *infobesity* or *information overload*. People are flooded by a continuous stream of information on the Internet.
In this work, the focus is on Twitter[1], the microblogging service and social network. On Twitter, users can publish short posts (called *tweets*) of up to 140 characters. The followers then receive the tweet.
We want to predict interest of users for the tweets they receive. This problem can be thought of as binary classification based on interest indicators. However, short text classification is a challenging task. The normal bag-of-words approach would not perform well on tweets because of the shortness. This work explores a semantic approach at feature generation for short text classification. We introduce a distinction between two kinds of feature generation methods, *shallow enrichment* and *deep enrichment*.
The contributions of this work are: (1) a method to finds entities mentioned in tweets using DBpedia, (2) a strategy to generate more relevant features for classification using deep enrichment (Section 3). DBpedia contains useful structured information that has already been extracted from Wikipedia. This structured information on DBpedia can be used directly in entity linking and feature generation. The main goal of this work is to show that deep enrichment using this information improves classification results w.r.t. shallow enrichment.
The rest of the paper is structured as follows. A review of relevant literature (Section 2) follows after this section. In Section 3, we focus on the classification part of the problem and define the shallow and deep enrichment methods, which are explained in detail in Section 4. The work is concluded with an experiment (Section 5) and a reflection giving a conclusion and discussing future work (Section 6).
The hypothesis this work will attempt to answer is whether deep enrichment using DBpedia gives better results than shallow enrichment. To accomplish this, we need to find a way to do deep enrichment and build (or reuse) a system that can extract entities from text.

[1] `www.twitter.com`

2 Related Work

First, we take a look at related short text classification and clustering methods. Then, the relevance of the Semantic Web for our work is explained, highlighting the DBpedia project. After that, we discuss some entity linking methods, and finally, semantic relatedness is discussed.

2.1 Short text classification and clustering

The main challenge of this work is to classify short text (tweets). A retweet prediction system based on Factorization Machines (FM) was proposed by Hong et al. [4]. Phan et al. [12] proposed a short text classification framework based on LDA with Wikipedia.
All of these differ significantly from the approach that is taken in this work. Here, it is attempted to do enrichment with DBpedia entities, linking the tweet to DBpedia. Using information directly from Wikipedia is found in some short text clustering approaches. One of these approaches is that of Banerjee et al. [1], who demonstrated that the use of Wikipedia page titles improves clustering accuracy, tested on news items from Google News. They constructed an Apache Lucene index of all Wikipedia articles. This index is used to answer two kinds of search queries based on the news items: (1) the title of the news item as the query and (2) the short description of the news item as the query.
A short text clustering approach that architecturally lies close to our approach was described by Xia Hu et al. [5]. They proposed a short text clustering framework that uses internal semantics extracted from the short text and external semantics obtained from world knowledge sources to do better clustering. Their framework consists of three major steps, (1) Hierarchical Resolution, where a three-level hierarchy of the short text is constructed, (2), Feature Generation, where the output of the previous step is used to search Wikipedia and Wordnet and (3), Feature Selection, where external features are filtered.

2.2 DBpedia

In this work, a semantic approach for classifying short texts is taken. We try to link the words and phrases in the tweet to named entities and use this information to improve classification. To accomplish this, a knowledge base is needed. The Semantic Web can be used for this purpose.
The **DBpedia** project [2] extracts structured information from Wikipedia. It contains entities that correspond to Wikipedia pages and relations between DBpedia entities and other resources, including other knowledge bases. DBpedia is the source of knowledge for our work.

2.3 Entity Linking

In this section, some entity linking systems are discussed. We start with Wikification.
The term Wikification was coined by Mihalcea and Csomai in their work on **Wikify!** [9]. They performed unsupervised keyword extraction that consists of two steps: (1) candidate extraction (find possible meanings of a term) and (2) keyword ranking. Mihalcea and Csomai concluded that keyphraseness[2] of a term outperforms the other keyword ranking criteria. In Wikify!, two orthogonal word sense disambiguation approaches are explored, both heavily relying on Wikipedia content.
Medelyan et al. [6] explored a topic indexing method using Wikipedia. The simple, unsupervised disambiguation approach described in the work of Medelyan et al. [6] is important for this work. Disambiguation scores are obtained by multiplying commonness[3] and relatedness[4].
Building on this, **Milne and Witten** [11] propose a new wikification system. It uses the semantic relatedness measure they described earlier [10] and takes a slightly different approach than earlier systems. Milne and Witten identify three important factors in disambiguation: (1) commonness, (2) relatedness, (3) context quality. Like in Medelyan et al. [6], Milne and Witten attempt to balance the commonness using context

[2] probability that a term is used to refer to some entity
[3] a priori chance that some string (called *surface form*) refers to a certain entity
[4] a score indicated how strongly related two entities or terms are

information. In contrast to Medelyan et al. [6], they train a classifier using these three features to learn a disambiguator. Link detection follows after disambiguation and is also based on machine learning. One of the features used in link detection is the disambiguation confidence from the disambiguation step, which explains why disambiguation is performed first.
An entity linking system for tweets with Wikipedia was described by Meij et al. [7]. They extracted many different features from short text and Wikipedia pages and used supervised machine learning techniques to learn to select best candidates for N-grams.
Linking to DBpedia is the approach taken in this work. A system that links regular text to DBpedia is **DBpedia Spotlight** [8]. Identification and disambiguation of entities takes a central role in Spotlight. First, N-grams are *spotted* that might refer to entities, then the DBpedia Lexicalization dataset is used to generate candidate entities for the spotted N-grams. Finally, in the disambiguation step, the best candidates are chosen. Disambiguation relies heavily on Wikipedia content, unlike the system by Milne and Witten [11] that purely relies on shared inlinks.

2.4 Semantic Relatedness

A semantic relatedness measure is important for word sense disambiguation. It approximates how strongly two terms or entities are related, based on their (possible) meanings. The systems described above differ in their approach to word sense disambiguation and semantic relatedness computation.
The Wikipedia linking system by **Milne and Witten** [11] uses a link-based measure the authors describe in their earlier work [10]. This measure is also used by Medelyan et al. [6]. Milne and Witten argue that their WLM (Wikipedia Link-based measure) is both cheaper and more accurate than ESA [3]. Earlier work by **Strube and Ponzetto** performs a broader study of semantic relatedness measures for **WikiRelate!** [13]. They define a categorization of such measures in three categories, (1) path based measures , (2) measures based on information content and (3) measures based on text overlap

3 Interest prediction on Twitter

Prediction of user interests for a new tweet can be done based on previous behavior of the user. What the user tweets about, what he/she retweets and favours are considered indicative of the user's interests. User interests are modeled by DBpedia entities, so when a user shows interest in a tweet, the user is assumed to be interested in at least one of the entities mentioned in the tweet. We formalize the classification problem below.

3.1 Problem Setting

Consider some Twitter user U. The Twitter feed $feed(U)$ of user U is a list of all tweets U receives from other Twitter users U is subscribed to and U's own tweets. Some tweets in $feed(U)$ originate from U, some tweets are retweeted by U and some other tweets are favoured by U. These three sets together are the interesting set $interest(U)$ for user U. All the other tweets, $feed(U) \setminus interest(U)$ form the neutral set of tweets, $neutral(U)$.
The classification problem can then be formalized as follows:

- Given: a user U and the user's Twitter feed $feed(U) = interest(U) \cup neutral(U)$
- Classify a new tweet not yet in $feed(U)$ in either $interest(U)$ or $neutral(U)$

This classification problem can be solved using a binary classifier, where $interest(U)$ are the positive examples and $neutral(U)$ the negative ones.

3.2 Feature generation for tweet classification

A baseline approach for text classification is the BOW approach with Naive Bayes. However, because tweets are very short and thus provide much less information than normal texts, we do not expect the BOW approach to provide good results. BOW+Naive Bayes had poor performance on the final dataset (as described below), finding only few true positives (average of 3 true positives out of the average 109 positively labelled tweets). To do better classification of short texts, a semantic approach is taken. Tweets are enriched with DBpedia entities, inspired by short text clustering approaches of Banerjee et al. [1] and Xia Hu et al. [5], who enriched short text using Wikipedia pages. Here, the tweets are linked to DBpedia entities, which can be used as features for classification. However, instead of just trying to link the tweet text to DBpedia (*shallow enrichment*), we also explore a second approach that adds additional DBpedia entities that are not directly observable in the tweet (*deep enrichment*).

3.2.1 Shallow enrichment

This method only enriches tweets with directly observable DBpedia entities. As an example, consider the following tweet:

"VIDEO: michael schumacher's last race"

With shallow enrichment, only the entities `dbpedia:Video`[5], `dbpedia:Michael_Schumacher` and `dbpedia:Racing` would be added as extra features for classification.
However, the classifier learning only from directly observable entities would be unable to use hidden entities that might better characterize the interest of the users. Continuing with the example, suppose the user is interested in Formula One. This user might show interest in several tweets about Michael Schumacher. The classifier would learn that this user is interested in Michael Schumacher. Now suppose the user receives a tweet about Sebastian Vettel (a colleague of Schumacher's), but the user has not received any tweets about Vettel before. So the classifier has never seen Vettel and would predict that this tweet about Sebastian Vettel is not interesting. A solution to this problem is to use deep enrichment.

3.2.2 Deep enrichment

This enrichment strategy goes one step further than shallow enrichment. The purpose of deep enrichment is to generate more features for classification on top of the entities found by shallow enrichment. This could be done by selecting certain properties of directly observable entities. However, to limit noise, only the most descriptive properties of the directly observable entities should be added as features to the tweet. Most descriptive properties of an entity are considered entities that are directly associated with the parent entity.
For example, a short description of Michael Schumacher is that he's a German Formula One racing driver. The task of deep enrichment is to find the entities that a directly observable entity is best associated with. For `dbpedia:Michael_Schumacher`, these might be `dbpedia:Germans`, `dbpedia:Formula_One` and `dbpedia:Auto_racing`. Adding these entities as features for classification can solve the problem of shallow enrichment. In the example about Michael Schumacher and Sebastian Vettel, the classifier would not only learn that tweets about Michael Schumacher are interesting, but also tweets about Formula One, racing, ... Deep enrichment on the tweet about Sebastian Vettel would also produce Formula One as a feature and the tweet will gain in interestingness from the perspective of the classifier.
Simply extracting entities from the short description text of some directly observable entity gives the desired most descriptive entities of that entity. The proposed deep enrichment method is in a sense "enriching the enrichment" obtained by shallow enrichment. The enrichment system used to enrich tweet texts can also be used to enrich a short entity description. The short description of an entity is obtained from the `rdfs:comment` property value of that entity.

[5] `dbpedia:<X>` stands for the named entity with URL http://dbpedia.org/resource/<X>

4 Semantic enrichment with DBpedia

DBpedia Spotlight [8] could be used to find DBpedia entities mentioned in short text. However, we chose to build a special enrichment system that is tailored for tweets. The new short text enrichment system is simple and uses (almost) exclusively information from DBpedia.
First, a short overview of the tweet enrichment system is given, followed by more detailed descriptions of the steps. Finally, the enrichment system used to enrich entities in deep enrichment is described.

4.1 Overview

The enrichment process is a pipeline consisting of four steps: (1) creating the N-gram hierarchy, (2) generating candidates, (3) computing relatedness scores and (4) merging the N-gram hierarchy. In the following subsections, these steps are described in detail. Two different enrichment pipelines can be distinguished in the enrichment system. The first one, the *tweet enrichment pipeline* is used to perform shallow enrichment on a tweet. The second pipeline is the *tagline enrichment pipeline* that is used to find the most descriptive entities of a DBpedia entity (for deep enrichment). Both pipelines are used in deep enrichment. In the last subsection of this section, the differences between the two pipelines are given.

4.2 Constructing the word N-gram hierarchy

This step is analogous to the Hierarchical Resolution step in the clustering framework of Xia Hu et al. [5] and the spotting stage of DBpedia Spotlight [8]. The purpose of this step is to generate a pruned hierarchy of word N-grams from the tweet text. Constructing the hierarchy consists of three steps, (1) removing Twitter-related syntax (RT, mentions, hashtags, cashtags), (2) generating N-gramming zones and (3) building the hierarchy from the N-gramming zones.
The first step is an obvious preprocessing step. The two following steps actually construct the pruned N-gram hierarchy. If an N-gram B is part of a larger N-gram A then B is a child of A in the hierarchy. Limiting the number of generated N-grams (pruning) is accomplished by first splitting the whole text in zones according to some simple rules and then building N-grams from these zones. This way, we get a (implicitly) pruned hierarchy.
For the Schumacher example (see above), some of the N-grams are: ”VIDEO”, ”schumacher”, ”race”.

4.3 Candidate Generation

This step corresponds to the Feature Generation step in [5], the candidate generation step in DBpedia Spotlight [8] and the candidate extraction step in Wikify! [9]. The purpose of this step is to find possible meanings (senses) for each N-gram constructed in the previous step.
Candidate generation is done by searching an Apache Lucene index using N-gram texts. There are two options for candidate generation indices. The first option is using a **Title Index**, which is built from entity names. A problem with this option is that it does not provide commonness information that might be useful for disambiguation in the Merging step of the pipeline. The second option involves a more sophisticated **Lexicalization Index**, which is build from the DBpedia Lexicalization dataset. This index consists of documents containing a *surface form*, a DBpedia entity URL and a commonness score of this combination.
A Twitter-specific addition is implemented in this step. Concatenated words often found on Twitter (e.g. hashtags) might contain useful information. To use this information, unigrams that initially do not give results are *shattered* in different ways and these shatterings are used to search the candidate generation index again. Shattering a unigram is simply inserting one or more whitespaces at different places. Only shatterings that give a full match are allowed as candidates. Partial matches of pieces of the shattering are not used.
For ”schumacher”, we may find senses `dbpedia:Michael_Schumacher`, `dbpedia:Schumacher_(crater)`,... where the first meaning has the highest commonness score.

4.4 Relatedness computation

This step is strongly inspired by WLM [10]. Milne and Witten [11] and Medelyan et al. [6] use the inlink-based semantic relatedness measure to disambiguate. In the Disambiguation step of Milne and Witten [11], relatedness of different senses of ambiguous terms is computed using unambiguous context terms. However, in tweets, one can not rely on existence of unambiguous terms. Therefore in this step of the enrichment pipeline, a simple, joint inlink-based relatedness computation algorithm is used that considers all possible contexts.

The joint inlink-based relatedness computation algorithm queries for entities linking to any of the entities that are candidate meanings of all N-grams of the tweet. This gives us the set I(nlinks). Then, for each entity $l \in I$ it retrieves the set of entities E_l that entity l links to. Each entity in I has at least one such entity (so $|E_l| > 0$). The relatedness scores of all entities in E_l are increased if: (1) they are not candidate meanings of the same N-gram, (2) they are not all the same and (3) the largest N-gram of all N-grams which have candidate meanings in E_l is not the parent of all the other N-grams that have candidate meanings E_l.

The entity `dbpedia:Michael_Schumacher` is related to `dbpedia:Auto_racing` (which might have been a sense of "race") but much less related to `dbpedia:Video`.

4.5 Merging

The final step in the pipeline collapses the N-gram hierarchy constructed in the first step to find the entities that are mentioned in the tweet. When collapsing the N-gram hierarchy, two choices can be distinguished: (1) Horizontal Choice and (2) Vertical Choice.

The **Horizontal Choice** determines the best candidate entity of an N-gram. This is disambiguation. We try to improve upon a simple multiplication of commonness and relatedness used by Medelyan et al. [6] to compute disambiguation scores by reducing the influence of commonness with growing relatedness scores.

The purpose of the **Vertical Choice** is to determine whether the meaning of a bigger (parent) N-gram will be used as a feature or the meaning(s) of (one of) its children. In some linking systems, this choice is part of choosing which parts of text are annotated. Often, this is done in the very beginning of the linking process, for example based on keyphraseness of an N-gram. Here, the vertical choice is deliberately made in the end because relatedness and disambiguation information may improve the decision. The decision is currently made using a simple formula that gives preference to longer N-grams, N-grams with a larger disambiguation score of their best candidate and N-grams with higher total relatedness percentage in the whole text.

In the Schumacher example, the string "last race" may refer to the film The Last Race (if it were on DBpedia), but its child "race" (with meaning `dbpedia:Auto_Racing`) should be chosen instead.

Finally, the entities from the collapsed N-gram hierarchy can be collected and filtered. **Filtering** of entities is based on their `rdf:type` property. Only specific entities are accepted (locations, places, organizations) as extra features for classification.

4.6 Tagline enrichment pipeline

The tagline enrichment pipeline uses the same steps as the tweet enrichment pipeline, but differs in the implementation of the relatedness computation step. **Relatedness computation** for tagline enrichment does not use the joint inlink-based algorithm. Instead, it uses the WLM [10] formula for semantic relatedness.

The semantic relatedness of a sense of an N-gram is computed with respect to the (parent) entity which is described by the tagline. This entity is the unambiguous context and is used in the same way as in Milne and Witten [11]. Suppose the tagline describes entity p, then the relatedness score of some candidate entity c is given by:

$$rel(c) = 1 - \frac{log(max(|P|, |C|)) - log(|P \cap C|)}{log(|W|) - log(min(|P|, |C|))} \quad (1)$$

where P and C are the entities containing pagelinks to p resp. c. $|W|$ should be taken equal to the number of entities in the database or another large number approximating this.

	Spotlight		Title-based		Lexicalized	
	shallow	deep	shallow	deep	shallow	deep
Accuracy	0.903	0.912	0.921	0.931	0.906	0.919
Precision	0.864	0.869	0.968	0.937	0.877	0.871
Recall	0.195	0.274	0.313	0.435	0.212	0.330
F-score	**0.302**	**0.403**	**0.448**	**0.575**	**0.318**	**0.467**
Specificity	0.996	0.993	0.998	0.994	0.996	0.993

Table 1: Combined results of first four volunteers

5 Experiments

The datasets used for classification evaluation are manual labellings of a list of tweets by 5 volunteers. To construct the datasets, first, 1000 consecutive tweets from @BBCWorld were taken at one point in time. These tweets were given to the volunteers who were instructed to label them as interesting or not, according to their *interests*. However, they were instructed to be *consistent*, that is, if they marked a tweet interesting that is clearly about some entity, they should mark all other tweets about that entity as interesting, unless they find them not interesting for an explicit reason[6].
5-fold interlaced cross-validation and the Naive Bayes classifier were used for evaluation. Three tweet enrichers were used, each in two versions: shallow and deep. The deep version constitutes the tweet text enricher (which is the shallow version), followed by tagline enrichment. The first enricher is based on DBpedia Spotlight. The two other versions are variations of our enrichment system. The main difference between the two is that the **Title-based** version does not use the Lexicalization index for candidate generation (and thus does not have commonness information), whereas the **Lexicalized** version is less up-to-date than the Title-based version. For each configuration, the standard evaluation metrics[7] were computed.

5.1 Experimental Results

The combined experimental classification results of the first 4 volunteers are shown in Table 1. The results of the fifth volunteer were not included because the algorithms performed so poorly that some evaluation metrics were not computable. Using the F-scores from Table 1 in a paired Student t-test for statistical significance of improvement resulted in at least 99% certainty that **deep enrichment improves upon shallow enrichment for entity-based interest classification of tweets**.

6 Conclusion and Future Work

Deep tweet enrichment gives better classification results than shallow enrichment. Deep enrichment extracts descriptive entities from the taglines of (observable) entities that were found by linking tweet texts to DBpedia and adds these descriptive entities as features (together with observable entities) whereas shallow enrichment only adds observable entities as features.
However, there are **limitations** on the interpretation of the results and the system. No tests were conducted to compare the tweet (and tagline) enrichers to other entity linking systems. This is required to draw solid conclusions whether and when our enrichment systems perform better. However, we could speculate that better classification results are associated with better linking performance.
The used datasets are taken at one point in time, from one news source and annotated consistently. Language usage of a news source is generally better than in an average tweet. Also, the system is currently not usable for interest prediction with real-world data in which interests are not consistently indicated. A limitation of the enrichment system is that it "thinks" in terms of (specific) entities only. The scope could be expanded to all DBpedia entities (not only entities from specific classes) by improving entity filtering. But even then,

[6] A reason that's not just "I already marked a tweet about this as interesting"
[7] TP, FP, TN, FN, recall, precision, F-score, specificity, accuracy

there is a fundamental limitation arising from the assumption that user interests can be modelled by entities. For example, some tweets may be interesting because they are funny. Users may also be interested in tweets of some source, regardless of the topic. Also, the Naive Bayes classifier limits the expressiveness of the system as it does not "understand" combinations of features. Enrichment could be improved by using machine learning for both Horizontal and Vertical Choices as well as for the final Filtering of entities.

References

[1] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering Short Texts using Wikipedia. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 787–788, New York, NY, USA, 2007. ACM.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A Crystallization Point for the Web of Data. *Web Semant.*, 7(3):154–165, Sept. 2009.

[3] E. Gabrilovich and S. Markovitch. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th international joint conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[4] L. Hong, A. S. Doumith, and B. D. Davison. Co-Factorization Machines: Modeling User Interests and Predicting Individual Decisions in Twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 557–566, New York, NY, USA, 2013. ACM.

[5] X. Hu, N. Sun, C. Zhang, and T.-S. Chua. Exploiting Internal and External Semantics for the Clustering of Short Texts using World Knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 919–928, New York, NY, USA, 2009. ACM.

[6] O. Medelyan, I. H. Witten, and D. Milne. Topic Indexing with Wikipedia. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence*, pages 19–24, Chicago, IL, USA, 2008.

[7] E. Meij, W. Weerkamp, and M. de Rijke. Adding Semantics to Microblog Posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 563–572, New York, NY, USA, 2012. ACM.

[8] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA, 2011. ACM.

[9] R. Mihalcea and A. Csomai. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM.

[10] D. Milne and I. H. Witten. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In *Proceedings of AAAI 2008 Workshop on Wikipedia and Artificial Intelligence*, pages 25–30, Chicago, IL, USA, 2008.

[11] D. Milne and I. H. Witten. Learning to Link with Wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA, 2008. ACM.

[12] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 91–100, New York, NY, USA, 2008. ACM.

[13] M. Strube and S. P. Ponzetto. WikiRelate! Computing Semantic Relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1419–1424. AAAI Press, 2006.